

Article 003 - Effective Techniques & Tools for Large Simulations - Part I

Contributed by Loren Abdulezer
Last Updated Friday, 29 June 2007

This is a multi-part article based on a webinar hosted by Decisioneering, Inc. on June 27th, 2007. The article outlines effective techniques and practices for managing large and complex spreadsheet simulations. Part I of the series deals with general spreadsheet construction techniques that become handy when working with large and complex spreadsheets.

I recently gave a Webinar/Presentation that was hosted by Decisioneering, the folks who make Crystal Ball. To supplement the PowerPoint slides and files available for download, I thought it would be instructive to provide more of a how-to explanation and/or discussion of the techniques.

This article is organized into four main topics:

- Spreadsheet techniques you might find useful but may not know about
- Techniques that may be helpful in your Crystal Ball simulations
- Constructing a Data Overpass
- An application of the Layered Approach design pattern

Some Useful Spreadsheet Techniques

{quotes}One of the interesting things about constructing spreadsheets is that there are few if any constraints placed on you as you build your spreadsheet models and simulations. This feature can be tremendously empowering or cobbling.{/quotes} It is all a matter of how you design and structure things. The very things that empower you for a small spreadsheet could be debilitating as your spreadsheet gets large or complex. The converse is true. Some techniques that appear ridiculously complicated and abstract for smaller sized spreadsheets can be tremendously liberating when you try tackling a difficult project.

I want to discuss some of these so-called "ridiculously complicated and abstract" techniques and show how they can be put to use.

Transforming a table into a linear list

Sometimes you may have a table of N rows and M columns and you would like to have it as a straight list (see Figure 1). It is not a big deal to have a formula that directly links the value in say, cell P9 to B9 or P20 to E11 when the table size is small. That is, the formula for P9 is

=B9

However when a table is large, this kind of direct table mapping is arduous and error prone. Obviously, there should be better ways to handle this.

I want to introduce an Excel function called OFFSET. At its simplest level offset starts returns the value of a spreadsheet cell based on

some starting point but shifted down a number of rows and shifted to the right a certain number of columns. OFFSET is great for looking up values in a table.

To look up values in a table say, Product C which is in the third row, and the West region which is in the fourth column to the right relative to the top left corner (cell A8 in Figure 1), we could use the formula:

```
=OFFSET(A8,3,4)
```

If you are working with a spreadsheet like the one shown in Figure 1, this formula would correctly retrieve the value 98. At first glance, it would appear that the OFFSET formula is just as tedious as directly mapping the table to the list one cell at a time. That is because the formula just illustrated uses hardwired numbers, like the values 3 and 4. If there were only a way to automatically generate all the row and column offsets, then we could pluck the values from the table. There are a number of ways to do this. I show you one way that uses a couple of hidden columns to cycle through the 1, 2, 3 for the products and 1, 2, 3, 4 for the geographic regions (see Figure 2).

Notice in columns K and L that all the cyclic combinations of the product offsets (1, 2, and 3) and the geographic region offsets (1, 2, 3, and 4) are permuted. It is easy enough to repeatedly enter the values 1, 2, 3 or whatever sequence fits the size of the table. The problem is, today the table is 3 by 4, six months from today it could easily be 30 rows by 40 columns, or even 3627 rows by 7 columns. You don't want to burden yourself with the task of typing in the numbers. It is much simpler to read off the dimensions of the table (cells G9 and G10 of Figure 2) and figure out what the row or column offset should be. There's an Excel MOD function (which stands for Modulo) that handles this quite well.

For the 12th item in the list, the row offset would be computed using a formula like:

```
=MOD($J$9-1,12)+1
```

In generality, if you had appropriately defined named ranges you could use a formula like:

```
=MOD(LinearListIndex-1,RowNumberInTable)+1
```

Having these row and column offsets are useful for another reason. In addition to placing the values of the table into a linear list, {quotes align=right}it is also important to create a compound label that appropriately identifies the corresponding row and column.{/quotes} You don't want to identify the value 51 as product in row offset 2 and region offset 1. You would want to use something like Product B:North.

Figure 3 summarizes some key formulas you may want in your hip pocket. It also identifies the spreadsheet file associated with this article that you can download.

In the Linearize.xls spreadsheet example, helper columns K and L are used to calculate the row and column offsets. Actually, the helper columns are not necessary. There's a technique called Inlining that accomplishes the incrementing without having to resort to the use of helper columns or rows. A summary of the formulas that would be used for inlining is shown in Figure 4.

The technique of Inlining is discussed in greater detail in my book Excel Best Practices for Business. The book also discusses in great detail how to use OFFSET.

Dynamic Sequences and Table Generation

Before I complete Part I of this article, I'd like to outline a clever technique for generating tables where the values in the table are determined contents of the row and column headers. As a simple example think of a multiplication table where the numbers 1 through 10 are in cells B1 through K1 and 1 through 5 are in cells A2 through A6. The formula in cell B2 that would compute the multiplied value of the respective row and column would be:

```
=$A2*B$1
```

You could replicate this formula in B2 all the way through K1 to complete a multiplication table. Setting up a table of this kind is not what I want to talk about. I have something else in mind that builds upon the table construction concept.

There's a clever technique I want you to think about. When you set up what I just described, you placed into the top row the sequence of the numbers 1, 2, 3, ... 10. That sequence could be any series of numbers of your choosing. You could use even numbers like 2, 4, 6, 8, 10, ... or odd numbers like 5, 7, 9, 11, 13, ...

Now, what about a sequence of quarter ending periods in months? (3, 6, 9, ...) or suppose you want to choose an interval like 3.75, 5, 6.25, 7.5, ... ? You could write a formula in cell C1 like:

```
=B1+1.25
```

and replicate it to the right. There's an annoyance with formulas of this kind. There are hardwired values in the formulas. Anytime the values like 1.25 need to be changed, you have to perform surgery on each and every cell where the hardwired value occurs. This is laborious and unnecessary.

There's a simpler technique. You only need to specify the first two numbers in the sequence. All the subsequent numbers can be automatically computed. Here is how you compute it. Multiply the second number in the sequence by two and then subtract the first number from it. Don't believe me? Try it. Using the last sequence, multiply 5 times two, which is 10. Now subtract 3.75 from this and you get 6.25. For the next pair of numbers in the sequence, multiply 6.25 times two to get 12.5. Subtract 5 from it and you have 7.5.

If 3.75 is in cell B1, 5 is in cell C1, then cell D1 could contain the formula:

```
=2*C1-B1
```

You can replicate this formula all the way to the right. Now whenever you adjust the first two numbers, all the remaining numbers will follow that sequence. It's not even necessary to set the values of the first two numbers by hand. They could be determined from a formula. That formula could have a random number in it, or be based on the day of the week, or what ever you care to use.

I provide a sample file to download with this article and summarize the formula in Figure 5.

This technique is great for making matrix style calculators (see my article in the Xcelsius Journal on how this is done).

Closing Thoughts

While these techniques do come into play with Crystal Ball simulations, they have applicability for building all kinds of spreadsheets. In Part II of this multi-part article I outline techniques that are very specific to Crystal Ball. In particular, I show how to set up a simulation where activities not only vary in terms of amount and probability of success, but can vary in terms of when they occur in a timeline. This can be very relevant in terms of timing of events for tax computations.

Additionally, in Part II I discuss how to drastically reduce the number of forecast cells in a Crystal Ball simulation, without cramping your ability to analyze the numbers you're most interested in.

The files for June 27th, 2007 webinar on "Effective Techniques and Tools for Modeling Very Large Spreadsheet Simulations" can be downloaded by clicking the File Vault link at the top of this page and searching for Article 003 in issue No.

1 (Please note: You need to register and login to access files on the File Vault . Fortunately, registration is a one time process, and it's free!).

©2007 Evolving Technologies Corporation - all rights reserved.